Ontology-based Software Capability Container for RESTful APIs

Abdelhadi Belfadel, Emna Amdouni, Jannik Laval, Chantal Cherifi, Néjib Moalla

Ontology-based software capability container for RESTful APIs

Abdelhadi Belfadel, Emna Amdouni, Jannik Laval, Chantal Cherifi and Nejib Moalla
University of Lyon, University Lumiere Lyon 2
DISP Laboratory
Lyon, France
{abdelhadi.belfadel, emna.amdouni, jannik.laval, chantal.bonnercherifi, nejib.moalla}@univ-lyon2.fr

***Abstract*—Software reuse and REST-based Web Applications resulted from open initiatives become an interesting opportunity for companies to save effort, time and cost during the design and development of new business needs. Gather and qualify these services in a container helps to discover, match and reuse them in developing new business applications for companies. Our objective in this work is the design of a software capability container offering a wider view qualification for REST-based services. Moreover, we aim to enrich the designed container with semantic elements to ease the discovery and the selection of the qualified services. In this purpose, we define an ontology based on a proposed Enterprise Architecture Capability Profile offering a qualification covering business, operational and technical aspects for services. Ontologies are widely acknowledged as a means to specify explicitly the meaning of concepts in a domain of interest, and to facilitate consistent sharing of data and knowledge pertaining to them. The qualification profile is based on a proposed meta-model that helps to retrieve and gather initial requirements used to guide the development of existing REST-based Web Applications. Furthermore, a Framework is proposed to exploit the designed container in order to respond to users requirements for developing future business process and efficiently reuse the qualified services. Our contribution aims to upgrade technical components to the level of end-users requirements. This helps to accelerate business application development and improve the reuse and sustainability of existing services.**

***Index Terms*—software reuse, capability container, software capability, ontology, Restful API, enterprise architecture, TOGAF, SOA, sustainability, business process, requirement analysis.**

I. Introduction

Several open source REST-based Web Applications result from different initiatives as "Factories of the Future" from the European Union. On the one hand, these initiatives offer to Small and Medium-Sized Enterprises (SMEs) solutions to help them exploring new solutions, prototype new business needs, bring some evolution to their existing applications and add new functionalities quickly with lower cost. This helps to improve productivity, bring ideas to the market and increase their competitiveness. On the other hand these applications expose services through RESTful APIs offering to developers reliable functionalities which are easy to integrate. Therefore, many research works are motivated to alleviate the usage of these REST-based SOA Applications, by proposing methods and tools to allow developers or end-users to build new business applications by composing on existing APIs [7], [18]. This helps to reduce effort spent on coding and maintenance activities, and increase software quality [31].

As mentioned, SMEs need to stay competitive therefore the modeling and prototyping of new business processes are required. The design and modeling of new business processes helps also to modernize existing systems and create new collaboration scenarios with other partners. As SMEs usually operate under limited resources, thus one design possibility is to maximize the reuse of existing solutions and services. This helps to control cost and quality, and moreover realize a prototype quickly to evaluate if the designed solution fits the specified requirements. But open source applications face some limits as they lack of documentation. As for instance description of business vision or requirements that helped in the realization of these application, as business goals and objectives. This can make the features of these solutions overlooked by the public due to difficulties to discover, to qualify quantitatively and qualitatively for reuse ends. Moreover, while user requirements lie in the problem area, software and service requirements lie in the solution area [20], it means more efforts are required to translate the user needs into software requirements.

The objective of this work is to increase software reuse through a software capability container, which offers a complete qualification with an adequate level of quality of service in design time and at runtime. In other words, we aim to retrieve and gather initial requirements and architecture knowledge from existing services in a software capability container. This helps to facilitate the decision making when choosing the suitable services that fit the users' requirements. The expected results from this work is a meta-model for software qualification, allowing to produce a qualification ontology gathering the qualified solutions in a repository for a future exploitation. This will improve the integration and usage of REST based web application for prototyping new business needs by offering more visibility on existing features for reuse ends.

However, answering to our research problematic was not a trival task given that we faced several problems that we can summarize as follows: the first problem is how to identify initial requirements and architecture knowledge used to guide the development of existing REST-based web applications? The second problem is how to structure the gathered knowledge in

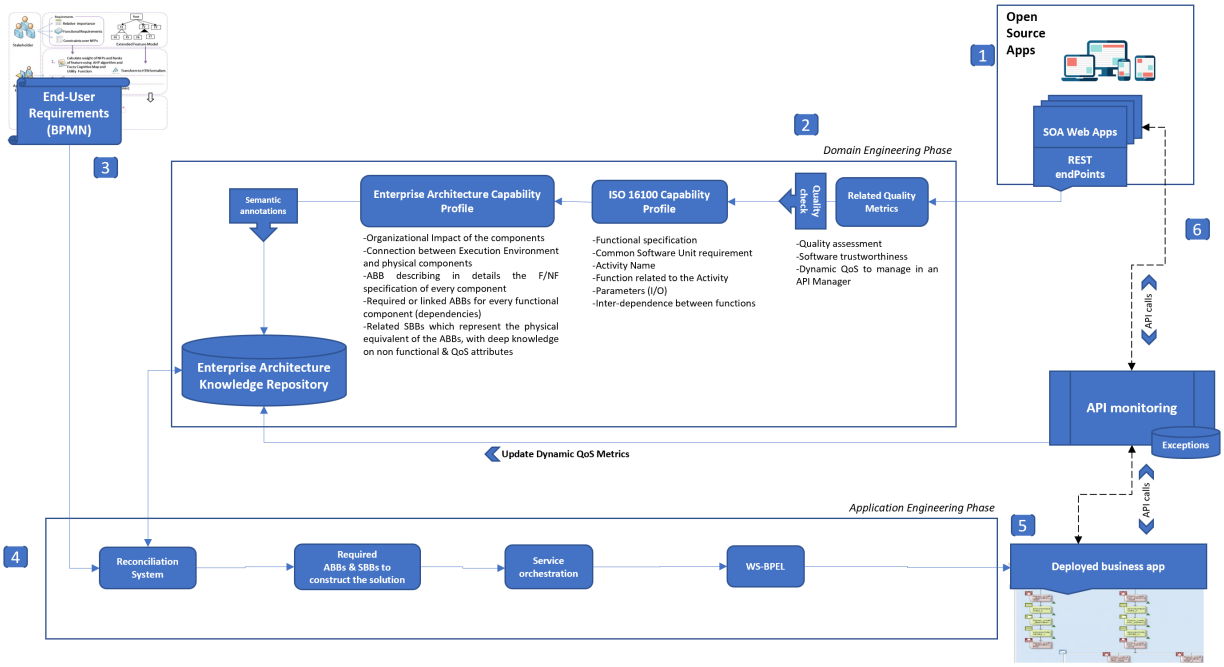

Fig. 1. Proposed Framework

different levels (i.e., operational, business and physical level)? The reformulated research problem is how to improve the reusability and sustainability of REST-based web applications to meet specific business needs?

In our previous works [6] and [5], we proposed a solution to respond to the specified research problem. We proposed an Enterprise Architecture Capability Profile (EACP) Framework that fits the problem and solution areas based on Enterprise Architecture Framework (TOGAF based) and Software Product Lines (SPL) paradigm. This helps to cover the two domains when qualifying an existing solution and offer higher-level of functional representations.

The EACP Framework proposed in our previous work is presented in Fig 1. Composed by two main phases as for the Software Product Lines (SPL) paradigm [17], [4]. The first phase refers to the domain engineering phase where quality check is applied to the REST-based Web Applications. This phase ends with a first qualification of the functional aspects based on the ISO 1610 norm. This first qualification level is enriched with the second qualification that gather the business, operational and non-functional specifications of the component. This final and complete qualification is stored in a container that we introduce under the name of Entreprise Architecture Knowledge Repository (EAKR) gathering all software capabilities as will be presented in this work. The result of this first phase is a connection between the REST based services and its related qualifications.

To maximize the reuse of the qualified components, end-users identify their requirements using BPMN notation.In the second phase of the EACP framework (Application Engineering Phase), the exploitation process is performed. This phase fetches and matches user requirements with existing and qualified software components in the Enterprise Architecture Knowledge Repository. The result of this phase is a set of Architectural Building Blocks (ABBs) and Solution Building Blocks (SBBs) that respond to the expressed needs. These SBBs are composed in order to generate a prototype to deploy and ready to be used by end-users.

The main part in the previous work concerns the EACP profile. It offers a complete qualification needed for exploring and reusing features when developing new business applications. This in an objective to maximize the reuse, low costs and adaptation effort. In the following, we describe the proposed qualification meta-model depicted in Fig. 2 and the relation between its different parts:

1) Organizational part: it is composed by the organization unit, with its related business goals and objectives.
2) Business part: it represents the new business process to realize. It is linked to a set of activities that composes the targeted business process to create. Added to this, this part describes the roles and actors that are concerned by the selected activities, and the event that could be trigged by an actor with a specific role.
3) Platform part: it describes the execution environments on which the targeted business application will run. This part is composed by the resulted application, related to the platform where the application will be deployed and the related components needed in the execution environment.
4) Architecture Builing Blocks (ABBs): it qualifies the existing feature by describing: (i) the business function, (ii) related attributes, (iii) corresponding constraints, (iv) the ability to communicate and exchange information,

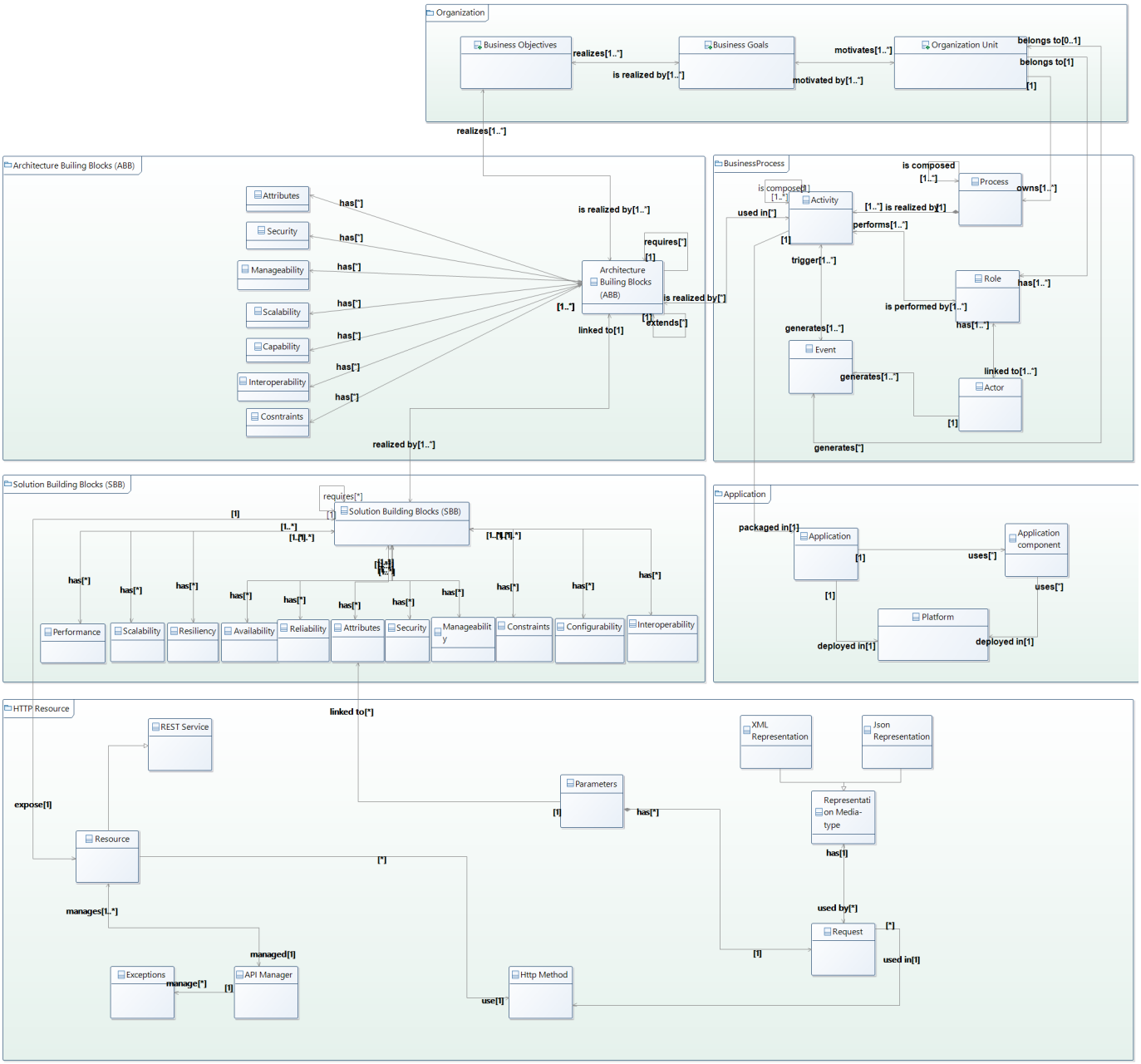

Fig. 2. Proposed meta-model

(v) the scalability that support the SOA web application that contains this feature, (vi) The manageability options offered by the application if they exist (for example, dedicated monitoring system), (vii) the security setup in this feature as authentication, authorization, input validations, data tempering, Distributed Denial of Service attack (DDoS) prevention. Added to this, it defines also (viii) the functional dependencies between features by descibing the required ABBs for every feature.

5) Solution Building Blocks (SBBs): it represents the physical equivalent of the ABBs. The SBB describes the technical specifications and quality of service (QoS). It has the same classes and attributes as for ABBs, but defines also dynamic quality attributes as availability metrics, resiliency and performance.
6) HTTP resource: it is exposed over the web and it represents the REST API giving access to the related SBB. It is defined by the URI, the HTTP method and the related parameters.

Each part of the meta-model is instantiated with an adequate level of details. Due to paper length constraints, we present in this work we present only the ABB model that is depicted in Fig.3.

The instantiated qualifications using the described model is

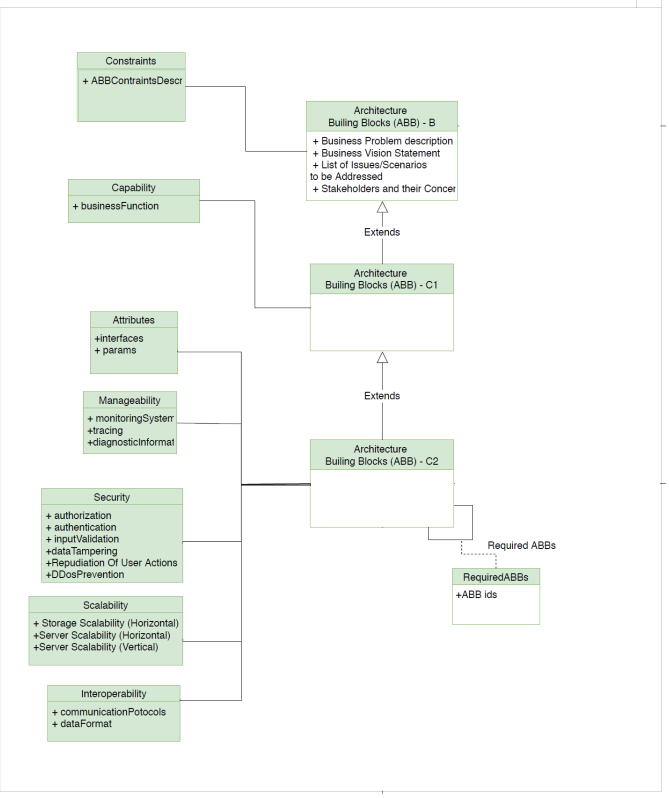

Fig. 3. Proposed Model for ABBs

saved in the EAKR container. The qualified services need to be enriched with semantic elements to ease the discovery and the selection of the most suitable service that fits the described user needs. The EAKR contains information that are associated to different levels (i.e., organization, business, functional, non-functional, http resource), and has a dynamic context with the continuous instantiation of the SBBs and related ABBs. Furthermore, we need a formal and explicit representation of these information. Due to these elements, we need an ontological modeling in order to construct this repository based on the meta-model and model already presented.

The main contribution of this work is: (i) the design of a wider view qualification for REST-based services, covering business, technical and organizational aspects; (ii) the proposal of a software capability container based on an ontology for service qualification.

The rest of this paper is organized as follows: Section II focuses on the Related work, and Section III presents the proposed qualification ontology. Section V presents the contribution and the targeted process for service discovery. Finally, conclusion and future work are drawn in Section VI.

II. Related work

As presented in the previous section, we need in this work to design a container of software capability by modeling the needed functional and non-functional aspects of REST-based services using an ontology. We believe that this will help to gather the desired qualification of REST-based services for discovery and reuse purpose.

Web service selection involves the discovery of services. It is the the process where the corresponding services meet the specific users' requirements [11]. Many researchers have proposed solutions to characterize services by adding non-functional attributes as Quality Of Service [12], [15], [25], or by adding semantics to functional attributes helping to realize ontology matching between requirements and services [10].

In the context of software engineering and software life cycle phases, there is a growing interest in the use of ontologies [8], [21]. In [9], the authors presented an ontology based on OWL-S that helps to describe quality properties defined in ISO 25010 and linked to software product. In [30] and [29], the authors proposed an ontology in Software Testing domain to organize software testing knowledge, and provide context-specific information to software testers. In [24], the authors proposed an automatic generation of software requirements in the form of an ontology using the UML diagrams. In [27], the authors presented usability requirement elicitation and specification method by using ontology. Our work in this context of software life cycle could be considered as input to the existing ontologies. As for testing domain, requirement elicitation or software quality, our work offers a complementary qualification for REST-based services to the existing ontological models.

In what follows, we focus on web service description used to design the proposed meta-model and the related ontology.

A. APIs and Web Service description

The description of an API is important for its successful adoption. APIs expose services and data and should be conceptualized and designed in an understandable way for consumers or developers, enabling them to start using it easily and quickly. APIs should provide an understandable and a humanreadable description. The API description is also parsed by machine to make server-side code or client stubs to consume the APIs. To make API documentation efficient, it has to include the following aspects as described in [14]: title, endpoint, method, URL parameters, message payload, header parameters, response code, error code. Many technologies and tools are available for API documentation as API Blueprint[1] or RESTful API Modeling Language (RAML)[2]. In our work, web service description is based on Open API Specification (known also as Swagger specification) designed for REST APIs.

Furthermore, several web service descriptions exist on literature and are based on the Web Service Description Language (WSDL) standard [13], which is often used in combination with SOAP and an XML Schema to provide Web services over the Internet. WSDL provides a machine-readable description, helping to know how the service can be called? what parameters does it expect? What response or data structures does it return? It specifies functional and non-functional properties

[1]https://apiblueprint.org/
[2]http://raml.org/

of Web services. The functional specifications describe inputs, outputs and operations exposed by a service by their name and data type. For the non-functional specifications, they describe the location of the service, the protocol used and the data type and format.

Since WSDL lacks the semantic needed to represent services capabilities, other languages using semantic are proposed by ontologies to allow reasoning on service capabilities as Ontology Web Language for Services (OWL-S) [19], WSDL-Semantic (WSDL-S) [1], Semantic Annotation for WSDL (SAWSDL) [16]. Other languages as Unified Service Description Language (USDL) [3] aims to provide a general framework for services and offer some features for instance support pre- and post-conditions.

B. Ontologies for service reuse

Business process management and SOA architectures has allowed the expansion of web service matchmaking and composition. In [10], the authors proposed a semantic profile covering the functional properties of Web Services that are used in a top-down approach for service matchmaking. Other projects as [26] and [22] focused on quality of service (QoS) to choose the suitable service which fits the non-functional properties. In [12] the authors proposed an ontology to qualify services from non-functional point of view. A matching between BPMN process and inputs and outputs of the services is realized, and a string comparison between name and data-type of parameters is proposed by taking into consideration the variations in QoS of services when selecting services that matches the user needs. In [23], authors propose a mechanism for web service selection based on input and output information over keyword based search.

C. Discussion

Most research works and practitioners used QoS for semantic web service selection [12], [15], [25] , by comparing and matching QoS attributes annotated in the web service description with the help of ontology using OWL-S or SAWSDL. So far, no effort has been made toward the web service discovery and selection that consider the business goals and objectives of the users' needs, nor the reconciliation with the initial requirements that helped the realization of the existing services.

The challenge in this case is to select the suitable REST based services that fit users' requirements, by taking into consideration : (i) The functional aspect as operational description, business need, organization of the users. (ii) Non-functional specifications as the technical descriptions with dynamic QoS in run-time or the platform where services are executed.

Our model shares some elements with OWL-S, and we need to reduce ambiguities in matching between users' requirements and features. Therefore we decided to extend OWL-S ontology based on existing ontologies. This helps to improve the coverage of REST-based services qualification. The next section describes the design process of the EACP Ontology and introduces our developed ontology.

III. EACP ONTOLOGY

Our developed ontology is composed of two main components: the terminological component (TBox) that describes in a formal way the related vocabulary of software capabilities and the assertion box (ABox) that stores ontology statements. To build our ontology, we considered the Uschold's methodology [28] and followed these steps: first, we defined the scope of the EACP ontology by formulating some competency questions (CQs) and extracting domain knowledge from the proposed meta-model (Fig. 2) and related model. Here, we cite only three CQS that we have identified:

- CQ_1: Retrieve ABBs from end-users' requirements based on business goals and objectives,
- CQ_2: Retrieve SBBs related to selected ABBs,
- CQ_3: Retrieve related SBBs that respect end-users' constraints.

Second, we identified existing ontologies that cover our domain needs and that are objective, modular, complete and coherent. We selected a top-level ontology called Basic Formal Ontology[3] (BFO) and two domain ontologies namely, OWL-S[4] and Information Artifact Ontology[5] (IAO). Third, we integrated and extended in a coherent way the selected ontologies into the EACP ontology using the Protégé Ontology Editor[6] V4.3. We note that only a subset of IAO was reused in our developed ontology. We excluded obsolete classes and classes that do not correspond to our scope. Finally, we evaluated the internal consistency and inferences of the model using the Fact++ reasoner.

Our resulted ontology contains 160 classes, 184 object properties and 33 data properties. EACP ontology is a modular ontology and it defines its classes and relations under the BFO framework. BFO is a widely used ontology its main goal is the integration of knowledge from different semantic resources and the modeling of data about entities in the world (here, we refer to our meta-model). We followed the alignment methodology that is described in [2] to map EACP to BFO. Based on our knowledge, the BFO ontology has not been used yet in the integration of service qualification meta-data. We resused IAO ontology to model data about information artifacts of the meta-model (e.g., conditional specification, objective specification, software application, etc.). In our work, we used the Web Ontology Language [7] (OWL) in the design of the TBox and the Resource Description Framework (RDF)[8] in the generation of the ABox's RDF triples.

Unlike the IAO ontology, OWL-S is not based on BFO. Thus, we realized a mapping work between IAO and OWL-S to integrate our three modules namely BFO, IAO and OWL-S into the EACP ontology. In Table II, we sumarize some top-level mappings between IAO and OWL-S to cover data of our

[3]http://basic-formal-ontology.org/
[4]https://www.w3.org/Submission/OWL-S/
[5]www.obofoundry.org/ontology/iao.html
[6]http://protege.stanford.edu
[7]https://www.w3.org/OWL/
[8]https://www.w3.org/RDF/

TABLE I
MAIN TOP-LEVEL CLASSES MAPPING TO ALIGN OWL-S TO IAO ONTOLOGY.WE NOTE THAT THE SYMBOL "$\subseteq$" DENOTES A SUBSUMPTION RELATION.

OWL-S class	ISA	Parent class IAO
OWL-S:service profile	$\subseteq$	IAO:directive information entity
OWL-S:service grounding	$\subseteq$	IAO:directive information entity
OWL-S:service model	$\subseteq$	IAO:directive information entity
OWL-S:service	$\subseteq$	IAO:software module
OWL-S:variable	$\subseteq$	IAO:information content entity
OWL-S:expression	$\subseteq$	IAO:information content entity
OWL-S:condition	$\subseteq$	IAO:conditional specification

meta-model. In the next paragraphs, *Italics* are used to denote semantic classes and relations of ontologies. Concerning the latter, labels are used (rather than actual IRI), for the sake of legibility.

The use of BFO classes and object properties hepled us in the integration of data and the formalization of knowledge. We used for example, the class *"BFO:information content entity"* to denote the entity "resource", the class *"BFO:material entity"* to represent the entity "organizational unit" and the class *"BFO:role"* to describe the entity "user role". We employed for example, the object property *"BFO:hasContinuantPartAtSomeTime"* to express that an *"EACP:architecture building block"* refers to at least one *"EACP:solution building block"* or the fact that an *"OWL-S:process"* has some *"OWL-S:output"*, etc.

Some pivotal classes of the EACP ontology are shown in Figure 4. As illustrated in this figure, UML entities "business goals" and "business objectives" are described as subclasses of the class *"IAO:objective specification"* via the use of the relation "IsA". The SSB is described as a subclass of *"IAO:software application"* and the ABB is defined as a subclass of *"IAO:software module"*. Table II illustrates some class mappings between our proposed meta-model and the EACP ontology.

We enriched the ontology with semantic axioms in order to improve it with reasoning capabilities. We added the following axioms:

- Necessary and sufficient conditions (i.e.,*"OWL-S:process" "OWL-S:atomic process"* or *"OWL-S:simple process"* or *"OWL-S:composite process"*);
- Existential restrictions as "some" restrictions (i.e., *OWL-S: process "BFO:hasContinuantPartAtSomeTime"* some *"OWL-S:condition"*);
- Universal restrictions as max restrictions (i.e., *"OWL-S:process" "OWL-s:name"* max 1 "Literal");
- Disjoint classes (i.e., *"OWL-S:input"* disjoint with *"OWL-S:output"*).

To conclude, EACP ontology has two main benefits: first, it is easily extensible to respond to new business needs thanks to its modular structure. Second, it insures the coherence of data even if we integrate other existing ontologies thanks to the BFO framework.

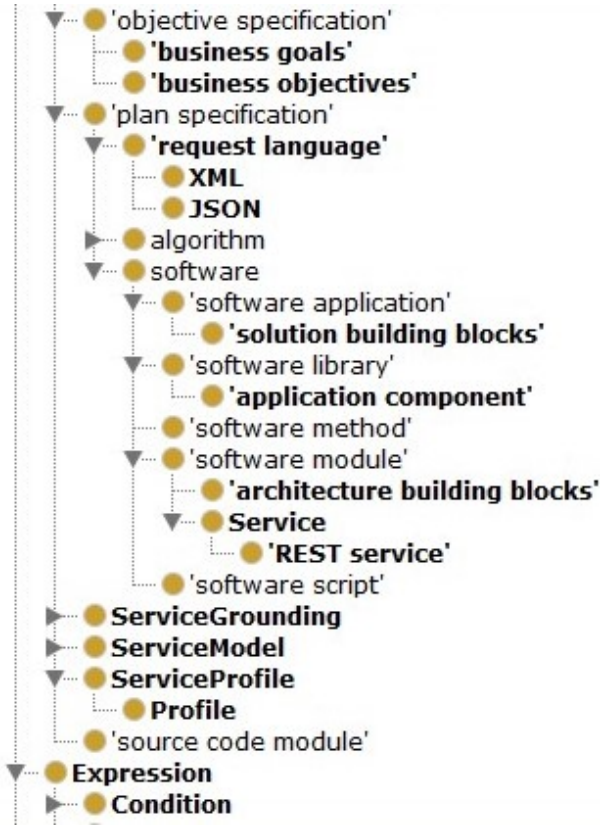

Fig. 4. Main classes of EACP ontology (Protégé Ontology Editor)

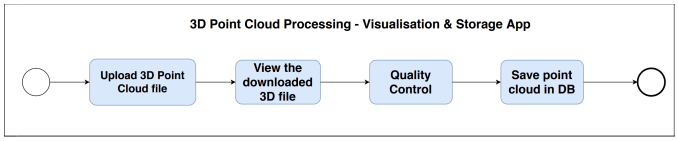

Fig. 5. 3D Point Cloud visualizer and storage app

IV. USE CASE

To validate our approach, we propose a concrete use case and an example of an instance of the EACP ontology.

The process as presented in Fig. 5 reflects an example of requirement sequence requested by an industrial end-user. The process aims to receive a 3D point could which represents a scanned piece. The received data are visualized using a 3D visualizer, and sent to a quality check which compares the received piece with a CAD design figure. Finally, the quality control report is saved in a database. The company has no knowledge about the existing services that could be reused, their quality or the relevance of a specific service between several possible solutions that can meet their specific needs.

As first action, we have already qualified as presented in section 1 a web application called 3DScan from FITMAN [9] project. A qualification of a REST service from this web application is realized using the proposed Enterprise Architecture Capability Profile and saved in the repository. In the following,

[9]www.fiware4industry.com

TABLE II
SOME CLASS MAPPINGS BETWEEN OUR PROPOSED META-MODEL AND THE EACP ONTOLOGY

Meta-model entity	EACP ontology class	Parent class in EACP ontology
Business objectives	EACP:business objectives	IAO:objective specification
Business goals	EACP:business goals	IAO:software module
Application	EACP:software application	IAO:software
Application component	ECAP:application component	IAO:software component
Process	OWL-S:process	OWL-S:service model
Role	BFO:role	BFO:realizable entity
Actor	EACP:actor	BFO:material entity
organizational unit	EACP:organizational unit	BFO:material entity
Architecture building blocks	EACP:architecture building blocks	OWL-S:software module
solution building blocks	EACP:solution building blocks	OWL-S:software module
Representation on media type	EACP:request language	IAO:plan specification
Parameters	OWL-S:parameters	IAO:information content entity

we present an instance of the ABB model as depicted in Fig. 3. This instance is saved in the Enterprise Application Knowledge Repository and exploited in the second phase of our proposed Framework (Application Engineering Phase).

Figure 6 illustrates an extract of the RDF graph of the ABB part concerning the web service *"get 3D object"*. We note that ABB's meta-data are stocked in an RDF/XML format. The pivotal instance in our graph is the ABB value that is annoted as an *"abbread3Dobject"*. This ABB value is contained in the description profile of the web service (i.e. *"profileRead3DObject"*) and has two main parts namely *"objectiveRead3DObject"* and *"goalsRead3DObject"*. We used the object property *"BFO:haspart"* to decribe the composition between two instances and the data property *"RDFS:label"* to decribe the textual content of instances (e.g., the *"goalsRead3DObject"* has as a label value the following content "achieve a high quality production without slowing down the production processes".).

Such an annotation work may help us to formalize and classify APIs' qualifications through i.e., the retrieval of ABB's from end-users requirements based on business goals and objectives or the selection of ABB's instances of a given SBB's parts.

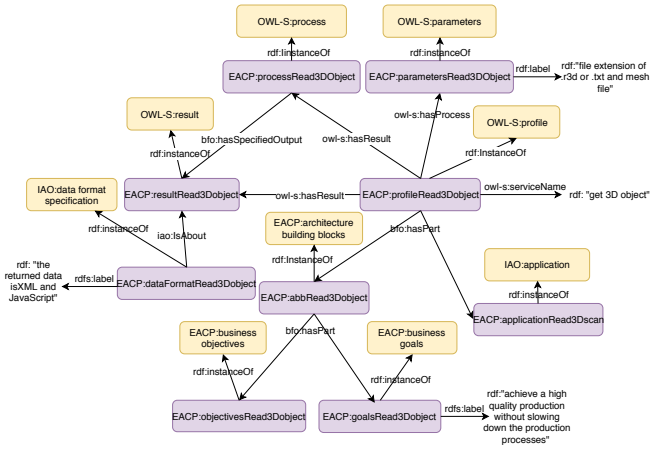

Fig. 6. EACP Ontology instance

V. CONTRIBUTIONS AND DISCUSSIONS

The main contributions of this work are: (i) the design of a wider view qualification for REST-based services, covering business, operational, technical and organizational aspects; (ii) the proposal of a software capability container based on EACP ontology for service discovery and orchestration; (iii) the proposal of a Framework that offers a complete qualification needed for exploring and reusing features when developing new business application; and (iv) upgrade technical components to the level of end-users requirements in order to accelerate business application development using service assets from existing SOA Web Applications.

The generated EACPOnto instances are saved in the Enterprise Application Knowledge Repository (EAKR) to be explored and exploited in the second phase of our proposed Framework (Application Engineering Phase) in Fig. 7. The process illustrates the actions that we aim to realize in order to exploit the EAKR. The end-user designs the required business process (in different level which concerns the design phase, implementation phase and deployment phase), with its organization constraints in BPMN format. We aim to evaluate as first action the business "wants" that must be delivered to meet the objectives and identify domain constraints. Then a reconciliation system based on Multi-criteria ABB selection between BPMN ontology and EACP Ontology will be proposed. This helps to match the suitable ABBs and related SBBs according to the defined business process and related constraints. A service orchestration is realized based on HTTP resources related to the SBBs in order to package the final application to the end-user and to deploy in the desired environment.

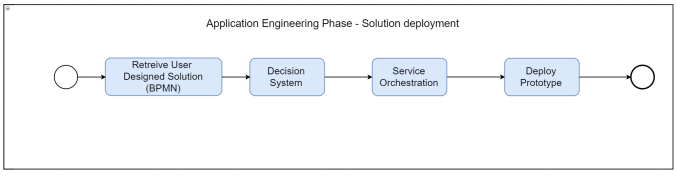

Fig. 7. Proposed Framework - Requirements Elicitation for the design phase

VI. Conclusion and future work

In this work, we propose a software capability container and its related ontology which offers a wider view qualification to improve the discovery and reuse of REST-based services. This container lies on a proposed Framework called EACP Framework and a meta-model based on TOGAF which is the most used enterprise architecture reference frame. The meta-model is also combined with dynamic quality metrics needed for services selection. As future work, we intend to propose a multi-criteria Architecture Building Blocks selection algorithm based on BPMN ontology and our proposed EACP Ontology. This helps to find and match services that fits the requirements expressed by end-users. Moreover, it aims to improve the reuse of REST-based Web Application for the implementation and deployment of new business process.

Acknowledgment

This paper presents some results that are developed in the Vf-OS project. This project has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement no. 723710. The content of this paper does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in this paper lies entirely with the authors.

References

[1] Rama Akkiraju, Joel Farrell, John A Miller, Meenakshi Nagarajan, Amit P Sheth, and Kunal Verma. Web service semantics-wsdl-s. 2005.

[2] Robert Arp, Barry Smith, and Andrew D Spear. *Building ontologies with basic formal ontology*. Mit Press, 2015.

[3] Alistair Barros and Daniel Oberle, editors. *Handbook of Service Description*. Springer US, 2012.

[4] Mahdi Bashari, Ebrahim Bagheri, and Weichang Du. Automated composition of service mashups through software product line engineering. In *International Conference on Software Reuse*, pages 20–38. Springer, 2016.

[5] A. Belfadel, J. Laval, C. B. Cherifi, and N. Moalla. Capability profile for enterprise application integration. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1333–1337, June 2017.

[6] Abdelhadi Belfadel, Jannik Laval, Chantal Cherifi, and Néjib Moalla. Towards service orchestration through software capability profile. In *I-ESA Interoperability for Enterprise Systems and Applications 2018*, Berlin, Germany, March 2018.

[7] Djamal Benslimane, Schahram Dustdar, and Amit Sheth. Services mashups: The new generation of web applications. *IEEE Internet Computing*, 12(5), 2008.

[8] MPS Bhatia, Akshi Kumar, and Rohit Beniwal. Ontologies for software engineering: Past, present and future. *Indian Journal of Science and Technology*, 9(9), 2016.

[9] María Julia Blas, Silvio Gonnet, and Horacio Leone. An ontology to document a quality scheme specification of a software product. *Expert Systems*, 34(5):e12213, 2017.

[10] Nicolas Boissel-Dallier, Frderick Benaben, Jean-Pierre Lorr, and Herv Pingaud. Mediation information system engineering based on hybrid service composition mechanism. *Journal of Systems and Software*, 108:39 – 59, 2015.

[11] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. W3c web services architecture, 2004.

[12] Sophea Chhun, Chantal Cherifi, Néjib Moalla, and Yacine Ouzrout. A multi-criteria service selection algorithm for business process requirements. *CoRR*, abs/1505.03998, 2015.

[13] Roberto Chinnici, Hugo Haas, Amelia A Lewis, Jean-Jacques Moreau, David Orchard, and Sanjiva Weerawarana. Web services description language (wsdl) version 2.0 part 2: Adjuncts. *W3C Recommendation*, 6, 2007.

[14] Brajesh De. *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Apress, 2017.

[15] Raluca Iordache and Florica Moldoveanu. Qos-aware web service semantic selection based on preferences. *Procedia Engineering*, 69:1152–1161, 2014.

[16] Jacek Kopeckỳ, Tomas Vitvar, Carine Bournez, and Joel Farrell. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6), 2007.

[17] J. Lee and G. Kotonya. Combining service-orientation with product line engineering. *IEEE Software*, 27(3):35–41, May 2010.

[18] Meherun Nesa Lucky, Marco Cremaschi, Barbara Lodigiani, Antonio Menolascina, and Flavio De Paoli. Enriching api descriptions by adding api profiles through semantic annotation. In *International Conference on Service-Oriented Computing*, pages 780–794. Springer, 2016.

[19] David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katia Sycara. Bringing semantics to web services: The owl-s approach. In Jorge Cardoso and Amit Sheth, editors, *Semantic Web Services and Web Process Composition*, pages 26–42, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[20] Pratap K.J. Mohapatra. *Software Engineering*. New Age International Pvt Ltd Publishers, 2009.

[21] Jeff Z Pan, Steffen Staab, Uwe Aßmann, Jürgen Ebert, and Yuting Zhao. *Ontology-driven software development*. Springer Science & Business Media, 2012.

[22] José Antonio Parejo, Sergio Segura, Pablo Fernandez, and Antonio Ruiz-Cortés. Qos-aware web services composition using grasp with path relinking. *Expert Syst. Appl.*, 41(9):4211–4223, July 2014.

[23] Lalit Purohit and Sandeep Kumar. Web service selection using semantic matching. In *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, page 16. ACM, 2016.

[24] Vahid Rastgoo, Monireh-Sadat Hosseini, and Esmaeil Kheirkhah. Semantic web-based software engineering by automated requirements ontology generation in soa. *International Journal of Web & Semantic Technology*, 5(2):1, 2014.

[25] Nanang Yudi Setiawan and Riyanarto Sarno. Multi-criteria decision making for selecting semantic web service considering variability and complexity trade-off. *Journal of Theoretical & Applied Information Technology*, 86(2), 2016.

[26] Toomas Tamm, Peter B. Seddon, Graeme G. Shanks, and Peter Reynolds. How does enterprise architecture add value to organisations? *CAIS*, 28:10, 2011.

[27] Chian Wen Too, Saadah Hassan, Abdul Azim Abdul Ghani, and Jamilah Din. Towards improving usability requirements elicitation and specification using ontology-driven approach. *Advanced Science Letters*, 23(5):4077–4081, 2017.

[28] Michael Uschold and Martin King. Towards a methodology for building ontologies. 1995.

[29] Shanmuganathan Vasanthapriyan, Jing Tian, and Jianwen Xiang. An ontology-based knowledge framework for software testing. In *International Symposium on Knowledge and Systems Sciences*, pages 212–226. Springer, 2017.

[30] Shanmuganathan Vasanthapriyan, Jing Tian, Dongdong Zhao, Shengwu Xiong, and Jianwen Xiang. An ontology-based knowledge sharing portal for software testing. In *Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on*, pages 472–479. IEEE, 2017.

[31] Minhaz F Zibran, Farjana Z Eishita, and Chanchal K Roy. Useful, but usable? factors affecting the usability of apis. In *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 151–155. IEEE, 2011.